Replication and Exploration of Generative Retrieval over Dynamic Corpora

Zhen Zhang
Shandong University
Qingdao, China
zhen.zhang.sdu@gmail.com

Xinyu Ma
Baidu Inc.
Beijing, China
xinyuma2016@gmail.com

Weiwei Sun
Carnegie Mellon University
Pittsburgh, USA
sunnweiwei@gmail.com

Pengjie Ren
Shandong University
Qingdao, China
renpengjie@sdu.edu.cn

Zhumin Chen
Shandong University
Qingdao, China
chenzhumin@sdu.edu.cn

Shuaiqiang Wang
Baidu Inc.
Beijing, China
shqiang.wang@gmail.com

Dawei Yin
Baidu Inc.
Beijing, China
yindawei@acm.org

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Zhaochun Ren*
Leiden University
Leiden, The Netherlands
z.ren@liacs.leidenuniv.nl

Abstract

Generative retrieval (GR) has emerged as a promising paradigm in information retrieval (IR). However, most existing GR models are developed and evaluated using a static document collection, and their performance in dynamic corpora where document collections evolve continuously is rarely studied. In this paper, we first reproduce and systematically evaluate various representative GR approaches over dynamic corpora. Through extensive experiments, we reveal that existing GR models with *text-based* docids show superior generalization to unseen documents. We observe that the more fine-grained the docid design in the GR model, the better its performance over dynamic corpora, surpassing BM25 and even being comparable to dense retrieval methods. While GR models with *numeric-based* docids show high efficiency, their performance drops significantly over dynamic corpora. Furthermore, our experiments find that the underperformance of numeric-based docids is partly due to their excessive tendency toward the initial document set, which likely results from overfitting on the training set. We then conduct an in-depth analysis of the best-performing GR methods. We identify three critical advantages of text-based docids in dynamic corpora: (i) Semantic alignment with language models' pretrained knowledge, (ii) Fine-grained docid design, and (iii) High lexical diversity. Building on these insights, we finally propose a novel multi-docid design that leverages both the efficiency of numeric-based docids and the effectiveness of text-based docids, achieving improved performance in dynamic corpus without requiring additional retraining. Our work offers empirical evidence for advancing GR methods over dynamic corpora and paves the way for developing more generalized yet efficient GR models in real-world search engines.

CCS Concepts

• **Information systems** → **Retrieval models and ranking**.

Keywords

Generative retrieval, Dense retrieval, Dynamic corpora

ACM Reference Format:
Zhen Zhang, Xinyu Ma, Weiwei Sun, Pengjie Ren, Zhumin Chen, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2025. Replication and Exploration of Generative Retrieval over Dynamic Corpora. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3730314

1 Introduction

Traditional information retrieval (IR) systems are based on pipelined "index-retrieval-then-rank" strategies where each module is optimized separately [6, 19, 22]. In recent years, a new retrieval paradigm, generative retrieval (GR) has emerged and garnered increasing attention due to its end-to-end training and remarkable performance [14, 24, 34, 37]. In this paradigm, all information of corpus is encoded into the parameters of a generative language model and such a model then directly generates document identifiers (docids) for a given query. Specifically, generative retrieval primarily consists of two steps: (i) *training as indexing*: through training objectives that map queries or documents to their associated docids, the model memorizes the document into its parameters during training and learns the mapping relationship between queries and docids. (ii) *generation as retrieval*: upon receiving a query, the model directly generates the corresponding docids autoregressively. Existing work has shown that the design of docid plays a critical role in the performance of GR models [28, 35, 38]. According to the type of docid, GR methods can be divided into two categories: *numeric-based* methods that convert documents into numeric sequences via quantization

*Corresponding author.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730314

strategies (hierarchical k-means [30], product quantization [36], residual quantization [33]), and *text-based* methods that leverage titles, URLs, or n-grams as docids [2, 5].

However, most GR approaches are built and evaluated in static scenarios, where the document collection remains fixed [9, 17, 29, 32]. Although these models perform strongly in such settings, their effectiveness in real-world environments, where documents continuously evolve over time, has not been thoroughly evaluated [3, 11]. Unlike static corpora, dynamic corpora impose higher demands on a model's generalization and robustness as new documents are not encountered during training. While previous studies have found that several GR models perform poorly over the dynamic corpora [3, 12], only limited efforts have been made to comprehensively investigate the generalization ability of different GR models under fair experimental settings.

An intuitive solution is to retrain the GR model from scratch when the underlying corpus is updated. However, the prohibitive computational costs of model training make this approach unfeasible [4]. Consequently, prior studies have explored continual (i.e., incremental) learning techniques to enhance the generalization ability of GR models with less computational overhead [7, 23]. For example, Mehta et al. [23] proposes DSI++, which employs sharpness-aware minimization to optimize for flat loss basins, thereby enabling the model to effectively learn from newly added documents. Guo et al. [7] develop task-specific adapters [8, 21] and pre-training objectives to adapt the dynamic corpora. As shown in Table 3, we find that SEAL [2], which employs text-based docids, can achieve comparable performance over dynamic corpora without re-training.

Therefore, our reproducibility study mainly focuses on the models' ability to generalize to unseen documents without additional training, and on analyzing the factors that influence the performance in dynamic corpora. Specifically, we first construct two dynamic corpora datasets based on the NQ [13] and MS-MARCO [1] datasets, by partitioning the entire document collection and user queries into several sets, thereby simulating practical dynamic retrieval scenarios. We then replicate various GR approaches and compare them with traditional retrieval models, including sparse retrieval and dense retrieval methods. Our findings show that existing GR models exhibit inconsistent performance over dynamic scenarios. In comparison, GR models that use text-based docids outperform those that employ numeric-based docids in terms of effectiveness, and underperform them in terms of efficiency.

Building on the insights listed above, we analyze the advantages of text-based docids and limitations of numeric-based docids over dynamic corpora:

- Models using numeric-based docids tend to generate docids that are encountered during initial training.
- Text-based docids maintain semantic alignment with language models' pretraining distribution, enabling better generalization with unseen documents.
- Text-based docids offer a finer-grained representation of documents, which enhances the model's ability to generalize to unseen documents.
- Text-based docids have higher lexical diversity, which helps mitigate overfitting to the initial training set.

Based on these observations, we design a novel multi-docid GR model that uses numeric-based docids but avoids the tendency to generate previously seen docids. It achieves finer granularity and larger docid size, and demonstrates superior performance and efficiency in dynamic corpora scenario.

In summary, our main contributions are as follows: (i) We introduce a comprehensive evaluation of existing GR approaches over dynamic corpora, focusing on their ability to generalize to unseen documents without additional training. (ii) We reveal the limitations of numeric-based docids in dynamic retrieval scenarios that they have a tendency toward the initial document set. (iii) We observed that the text-based docid performs better on dynamic corpora and analyzed key factors for its performance, including semantic alignment, fine-grained docids design, and high lexical diversity. (iv) Building on our findings, we propose a novel GR framework that integrates the high efficiency and low memory of numeric-based docids with the strong generalization capabilities of text-based docids.

2 Background and Preliminaries

2.1 Generative retrieval

The document retrieval task retrieves a relevant document d from a document collection $\mathcal{D}$ given a user query q. Traditional retrieval approaches typically rely on inverted indexing or similarity-based ranking; whereas GR retrieves documents by autoregressively generating the docid z corresponding to the most relevant document.

A GR model typically consists of two key components: an *indexer* and a *retriever*. The *Indexer* acts as a document encoder that maps each document $d \in \mathcal{D}$ to a unique identifier sequence (docid) $\mathbf{z} = \{z_1, z_2, \ldots, z_M\}$. Here each element $z_t \in [K]$ is drawn a predefined vocabulary V, which may include numerical tokens, lexical items, or other semantic identifiers. There are mainly two types of docid design: numeric-based and text-based:

- numeric-based: Each document corresponds to a numeric sequence, typically generated by converting document embeddings into numeric sequences using clustering or quantization methods.
- text-based: Uses metadata from documents as docid, generally including elements such as title, query, URL, n-gram, etc.

As for the *retriever*, upon taking a user query $q \in \mathcal{Q}$, the *Retriever* generates the most relevant docid by maximizing the conditional probability over the sequence of tokens representing the docid:

$$P(z_1, z_2, \ldots, z_M \mid q; \theta) = \prod_{t=1}^{M} P\big(z_t \mid z_1, \ldots, z_{t-1}, q; \theta\big), \tag{1}$$

where θ denotes the model parameters, M is the predefined docid length, and $\{z_1, \ldots, z_{t-1}\}$ represents the previously generated tokens. During inference, the system performs beam search to find:

$$\hat{\mathbf{z}} = \underset{\mathbf{z} \in \mathcal{T}_{\mathcal{D}}}{\operatorname{argmax}} P(z_{1:M} | q; \theta), \tag{2}$$

where a prefix tree ($\mathcal{T}_{\mathcal{D}}$) enforces constrained decoding by storing all valid docid prefixes from the corpora $\mathcal{D}$. At each decoding step t, token selection is restricted to the children nodes of the current prefix $z_{1:t-1}$ in $\mathcal{T}_{\mathcal{D}}$, ensuring generated sequences correspond to existing documents.

2.2 Task formulation

The dynamic corpora involves continuously adapting to an expanding document collection while maintaining retrieval capabilities. Initially, we are given:

- an initial document set $\mathcal{D}_0 = \{d_1, d_2, \ldots, d_n\}$, and
- corresponding query-document pairs $\mathcal{P}_0 = \{\langle q_i, d_j \rangle \mid q_i \in \mathcal{Q}, d_j \in \mathcal{D}_0\}$,

where each document $d_j \in \mathcal{D}_0$ is a text sequence and $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$ denotes the initial query set. A GR model is first trained using standard sequence-to-sequence objectives on $\mathcal{P}_0$. The core challenge arises with incremental updates: when new documents $\mathcal{D}_{\text{new}} = \{d_{n+1}, \ldots, d_{n+k}\}$ are introduced. For each new document $d_{n+i} \in \mathcal{D}_{\text{new}}$, the *indexer* generates a corresponding docid:

$$\mathbf{z}_{n+i} = \{z_1^{(n+i)}, \ldots, z_M^{(n+i)}\}, \quad z_t^{(n+i)} \in \{1, \ldots, K\}, \tag{3}$$

preserving the original token vocabulary and sequence length M. These new identifiers are incorporated into the existing prefix tree $\mathcal{T}_{\mathcal{D}}$.

The *retriever* must then handle the expanded search space $\mathcal{D}' = \mathcal{D}_0 \cup \mathcal{D}_{\text{new}}$. Two adaptation strategies emerge: (i) **Model adaptation**: Continuously train or retrain the model on $\mathcal{P}_0 \cup \mathcal{P}_{\text{new}}$ to learn updated document representations. (ii) **Index adaptation**: Maintain frozen model parameters while updating only the indexing structures. As mentioned in the Introduction Section, retraining or continuing training the GR model would cost much more computational overhead. Therefore, our work only focuses on the second paradigm, where the *Retriever* must leverage learned knowledge to handle novel documents through constrained decoding over the updated prefix tree $\mathcal{T}'_{\mathcal{D}}$. This setting tests the model's ability to generalize to unseen document representations without parameter updates.

2.3 Evaluation metrics

Our evaluation primarily focuses on two aspects of model performance over dynamic corpora: (i) *retrieve initial documents*, which assesses the model's ability to maintain performance on queries targeting original documents $\mathcal{D}_0$; and (ii) *retrieve newly added documents*, which evaluates the model's capacity to retrieve novel documents $\mathcal{D}_{\text{new}}$. We use `Hit@10` as the primary metric in both settings and further introduce formal metrics to summarize the model's performance as new documents are incrementally indexed.

Retrieve initial documents. To assess the model's forgetting behavior when indexing new documents, we define the forgetting metric F_n, which quantifies the degradation in retrieval performance on queries targeting the original corpus $\mathcal{D}_0$ after indexing corpus $\mathcal{D}_1$ to $\mathcal{D}_n$:

Definition 1 (Forgetting Metric F_n).

$$F_n = \frac{1}{n} \sum_{o=1}^{n} \max\left(P_{0,0} - P_{o,0},\ 0\right) \tag{4}$$

where $P_{o,0}$ represents the retrieval performance (e.g., Hit@10) of queries in $\mathcal{D}_0$ after indexing corpus $\mathcal{D}_o$.

Retrieve newly added documents. To measure the model's ability to generalize and retrieve newly indexed documents, we define the generalization performance metric GA_n, which captures how well the model retrieves queries associated with incrementally added documents:

Definition 2 (Generalization Performance GA_n).

$$GA_n = \frac{1}{n} \sum_{o=1}^{n} P_{o,o} \tag{5}$$

where $P_{o,o}$ represents the retrieval performance on queries targeting $\mathcal{D}_o$ after indexing corpus $\mathcal{D}_o$.

3 Experimental Setup

3.1 Datasets

We conduct our experiments on two widely used datasets: Natural Questions (NQ) [13] and MS-MARCO [1]. We appropriately partition the document sets within the datasets to simulate dynamic corpora scenarios.

To simulate the task of dynamic corpora, we partition the documents sets in MS-MARCO and NQ through a two-phase process: (i) **Initial corpus construction:** Collect all documents and randomly select 50% as the initial corpus $\mathcal{D}_0$. Extract query-document pairs associated with $\mathcal{D}_0$ and split them into a training set $\mathcal{P}_0$ and an initial test set $\mathcal{Q}_0$. All models are fully trained on $\mathcal{P}_0$, simulating initialization on the base corpus. (ii) **Incremental document additions:** Partition the remaining 50% of documents into five equally sized subsets, each comprising 10% of the original corpus ($\mathcal{D}_1$ to $\mathcal{D}_5$). For each $\mathcal{D}_i$ ($1 \le i \le 5$), extract its associated query-document pairs to form incremental test sets $\mathcal{Q}_1$ to $\mathcal{Q}_5$. No additional training is performed for these document sets—models only index new documents upon each addition.

3.2 Retrieval approaches

To compare the performance of GR approaches with previous retrieval methods on dynamic corpora, we selected three types of retrieval approaches: sparse retrieval approaches, dense retrieval approaches, and generative retrieval approaches.

Sparse retrieval. (i) BM25 [26], a traditional retrieval approach that ranks documents according to the frequency of the term and the normalization of the length of the document. We re-index the entire corpora when the corpora is updated.

Dense retrieval. (i) DPR [10], a neural retrieval method that uses dual encoders to map queries and documents into a shared dense vector space for similarity computation. We use the document encoder trained on $\mathcal{D}_0$ to re-encode the newly added documents for retrieval. (ii) DPR-HN [20, 25], an enhanced DPR variant that incorporates hard negative sampling techniques. It integrates in-batch negatives, top-K retrieved negatives from dense retrievers, and BM25 hard negatives.

Generative retrieval. (i) DSI-SE [30], which uses the hierarchical k-means clustering results of document embeddings as docids and trains the model to memorize the documents in parameters. (ii) Ultron-PQ [36] uses the product quantization results of document embeddings as docids and designs a three-stage training task to enable the model to memorize the documents. (iii) Ultron-URL [36], a variant of Ultron, uses the document URLs as docids. (iv) NCI [31] uses a prefix-aware weight-adaptive decoder and various query

Table 1: Model performance on queries corresponding to the initial document set $\mathcal{D}_0$, where the document set size continuously expands as new documents are added. The best results are indicated in boldface.

Method	DocID Type	NQ (Hit@10)							MS-MARCO (Hit@10)						
		$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$F_n \downarrow$	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$F_n \downarrow$
Sparse retrieval															
BM25	Term Weight	0.647	0.625	0.611	0.598	0.573	0.573	0.051	0.653	0.640	0.632	0.629	0.619	0.614	0.026
Dense retrieval															
DPR	Dense Vector	0.725	0.704	0.696	0.686	0.670	0.660	0.042	0.683	0.681	0.668	0.656	0.651	0.648	0.022
DPR-HN	Dense Vector	0.826	0.801	0.797	0.776	0.773	0.768	0.043	**0.723**	**0.712**	**0.692**	**0.685**	**0.672**	**0.664**	0.038
Generative retrieval															
DSI-SE	Category Nums	0.718	0.710	0.706	0.702	0.699	0.696	**0.015**	0.605	0.601	0.597	0.594	0.592	0.589	**0.010**
Ultron-PQ	Category Nums	0.795	0.785	0.780	0.780	0.762	0.755	0.023	0.663	0.655	0.647	0.643	0.637	0.632	0.020
NCI	Category Nums	**0.871**	0.856	0.844	**0.839**	0.811	0.802	0.041	0.702	0.693	0.673	0.667	0.654	0.633	0.038
GenRET	Category Nums	0.858	0.853	0.836	0.829	0.812	0.796	0.033	0.717	0.697	0.688	0.674	0.659	0.652	0.043
Ultron-URL	URL Path	0.816	0.810	0.794	0.781	0.780	0.768	0.029	0.626	0.620	0.618	0.614	0.611	0.608	0.012
SEAL	N-gram	0.809	0.806	0.788	0.774	0.774	0.763	0.028	0.661	0.641	0.625	0.616	0.602	0.598	0.045
MINDER	Multi-text	0.838	0.828	0.813	0.811	0.801	0.773	0.033	0.667	0.649	0.633	0.625	0.612	0.600	0.043
LTRGR	Multi-text	0.862	**0.857**	**0.846**	0.827	**0.813**	**0.807**	0.032	0.688	0.675	0.660	0.649	0.636	0.621	0.040

generation strategies to train the model. (v) GenRET [27] uses constrained cluster centroids as docids and trains the representations of docids through document tokenization and document reconstruction tasks. (vi) SEAL [2] uses arbitrary n-grams from documents as docids and generates a series of n-grams under the constraints of the FM-index to retrieve the corresponding documents. (vii) MINDER [15] adopts multiple text types to represent docids, such as titles, URLs, and n-grams. Different types of scores are generated at the same time and documents are retrieved based on these scores. (viii) LTRGR [16] also adopts the multiple text docid design, and introduces an additional learn-to-rank task and rank loss to optimize the retrieval model.

To enable GR approaches to handle dynamic corpora, we adopt the following design strategies for docid assignment: (i) **For numeric-based docids**: These approaches rely on structured numerical representations (e.g., k-means centroids, product quantization codebooks). We preserve the original document encoder's state (k-means cluster centroids or vector quantization codebooks) to encode new documents, maintaining identifier consistency with initial documents. (ii) **For text-based docids**: These approaches leverage document-induced textual patterns (e.g., Title, URLs). We leverage new documents' inherent metadata and text structure to automatically assemble valid identifiers.

3.3 Implementation details

We implement BM25 using the `bm25s` library.[1] For dense retrieval models (DPR and DPR-HN), we employ the `pyserini` toolkit [18], using its built-in functionalities for indexing and retrieval. For GR approaches (DSI, ULtron variants, NCI, GenRET, SEAL, MINDER, and LTRGR), we adopt their official implementations and strictly follow the default hyperparameter configurations provided in the original works to ensure reproducibility.

All models operate with a maximum input sequence length of 512 tokens. Experiments are conducted on 8 NVIDIA A100 GPUs, with distributed training enabled for GR methods to accommodate their large parameter sizes. For DPR/DPR-HN, we initialize the document encoder with the checkpoint pre-trained on the initial documents and freeze its parameters during incremental phases.

[1] https://github.com/xhluca/bm25s

3.4 Statistical validation

We verified the reliability of retrieval performance. All experimental results reported in Tables 1 and Table 2 achieved statistical significance at $p < 0.05$.

4 Performance over Dynamic Corpora

In this section, we analyze the experimental results in dynamic corpora scenario, focusing on how different retrieval approaches perform as the document set expands incrementally.

Retrieving initial documents. We investigate how the incremental indexing of new documents affects the retrieval performance of queries corresponding to the initial document set $\mathcal{D}_0$, as the document collection grows. Both BM25 and DPR exhibit stable performance in maintaining retrieval effectiveness for initial documents $\mathcal{D}_0$.

Generative retrieval methods demonstrate better resistance to forgetting. For example, DSI-SE achieves F_n values (0.015 on NQ and 0.010 on MS-MARCO), outperforming dense retrieval approaches. Because there is no additional training step, the mapping from documents to docids remains intact, enabling these models to maintain consistent performance for the initially indexed documents. However, a potential drawback lies in their limited flexibility when new documents are introduced, as these methods often struggle to index unseen docids effectively unless specific adaptations (e.g., additional training) are employed.

Retrieving newly added documents. When retrieving newly added documents from a dynamically expanding document collection, both BM25 and DPR demonstrate stable generalization performance (GA_n) when retrieving newly added documents, as measured by their ability to adapt to an expanding corpus. BM25

Table 2: Model performance on queries corresponding to the newly added document set $\mathcal{D}_1$ to $\mathcal{D}_5$, where the document set size continuously expands as new documents are added. The best results are indicated in boldface.

Method	DocID Type	NQ (Hit@10)							MS-MARCO (Hit@10)						
		$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$GA_n \uparrow$	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$GA_n \uparrow$
Sparse retrieval															
BM25	Term Weight	0.647	0.620	0.588	0.598	0.552	0.571	0.586	0.653	0.634	0.631	0.620	0.603	0.601	0.618
Dense retrieval															
DPR	Dense Vector	0.725	0.580	0.587	0.570	0.531	0.544	0.562	0.683	0.625	0.623	0.599	0.607	0.604	0.612
DPR-HN	Dense Vector	0.826	0.645	0.644	0.626	0.621	0.624	0.632	**0.723**	**0.662**	**0.653**	**0.642**	**0.623**	**0.619**	**0.640**
Generative retrieval															
DSI-SE	Category Nums	0.718	0.231	0.203	0.221	0.185	0.205	0.209	0.605	0.204	0.197	0.186	0.172	0.159	0.184
Ultron-PQ	Category Nums	0.795	0.548	0.549	0.542	0.539	0.532	0.542	0.663	0.428	0.415	0.399	0.384	0.376	0.400
NCI	Category Nums	**0.871**	0.464	0.437	0.433	0.358	0.323	0.403	0.702	0.402	0.380	0.355	0.341	0.320	0.360
GenRET	Category Nums	0.858	0.361	0.419	0.401	0.357	0.354	0.378	0.717	0.439	0.425	0.396	0.350	0.331	0.388
Ultron-URL	URL Path	0.816	0.553	0.545	0.543	0.541	0.532	0.543	0.626	0.397	0.376	0.364	0.354	0.342	0.367
SEAL	N-gram	0.809	0.744	0.736	0.727	0.727	0.725	0.732	0.661	0.611	0.607	0.584	0.571	0.559	0.586
MINDER	Multi-text	0.838	0.803	0.751	0.746	0.742	0.736	0.756	0.667	0.614	0.608	0.587	0.569	0.546	0.585
LTRGR	Multi-text	0.862	**0.831**	**0.803**	**0.811**	**0.779**	**0.773**	**0.799**	0.688	0.621	0.612	0.601	0.589	0.577	0.600

achieves GA_n scores of 0.586 on NQ and 0.618 on MS-MARCO, while DPR attains 0.562 and 0.612 respectively.

Generative retrieval models exhibit huge divergence in generalization performance (GA_n): Numeric-based docid (e.g., DSI-SE, Ultron-PQ) demonstrate critical limitations in adapting to unseen documents. DSI-SE's GA_n values (0.209 on NQ, 0.184 on MS-MARCO) align with its severe Hits@10 degradation when retrieving new documents (e.g., dropping from 0.718 to 0.205 on NQ's $\mathcal{D}_5$). This failure stems from rigid numeric docid mappings learned during training, which lack inherent semantic connections to new content. Similarly, Ultron-PQ's numeric identifiers yield unstable GA_n (0.542 on NQ, 0.400 on MS-MARCO), as its quantization-based docid system struggles to encode novel document semantics without retraining.

Text-based docid methods, such as SEAL and LTRGR, demonstrate much more promising results. For example, LTRGR attain $GA_n = 0.799$ on NQ, with only a 0.089 Hits@10 drop from $\mathcal{D}_0$ to $\mathcal{D}_5$, outperforming even optimized dense retrieval (DPR-HN: $GA_n = 0.632$). SEAL's n-gram docids similarly excel ($GA_n = 0.732$ on NQ). This suggests that text-based docids, such as n-grams or titles, offer more flexibility and generalization. Text-based docids can inherently adapt to newly added documents, which often contain similar linguistic features. Consequently, models using text-based docids are better equipped to maintain high retrieval performance as the corpus evolves. These models can leverage the semantic richness of text-based representations, allowing them to effectively handle the continuous expansion of the document set without significant performance loss.

Incremental training vs direct generalization. We implemented incremental training inspired by the DSI++ framework for both DSI and SEAL. Our approach involves two key components: (i) incremental indexing training, where models are fine-tuned on newly added documents to learn their docid mappings, and (ii) random replay training, which reintroduces a subset of historical documents during fine-tuning to mitigate forgetting. As shown in Table 3, incremental training significantly enhances the models' ability to retrieve newly added documents for DSI. For instance, DSI++ improves Hits@10 for new documents on NQ from 0.205 to 0.644 on $\mathcal{D}_5$, demonstrating better generalization to unseen documents. However, this comes at a cost: DSI++ exhibits noticeable forgetting on initial documents. For SEAL, incremental training yields only a slight improvement in both settings. This is because SEAL's text-based docids inherently accommodate semantic variations in new documents, reducing the need for extensive retraining. Further analysis in Table 3 suggests that while incremental training provides modest benefits for text-based docid models, their robustness to dynamic corpora primarily stems from their ability to generalize via natural language patterns, rather than relying on explicit retraining.

Table 3: Comparison of retrieval performance (Hit@10) on the NQ dataset for initial and newly added documents.

Model	NQ (Hit@10)					
	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$
Initial documents						
DSI	0.718	0.710	0.706	0.702	0.699	0.696
DSI++	0.718	0.697	0.687	0.682	0.676	0.673
SEAL	0.809	**0.806**	0.788	0.774	0.774	0.763
SEAL++	**0.809**	0.791	**0.788**	**0.781**	**0.776**	**0.766**
Newly added documents						
DSI	0.718	0.231	0.203	0.221	0.185	0.205
DSI++	0.718	0.677	0.671	0.667	0.657	0.644
SEAL	0.809	0.744	0.736	0.727	0.727	0.725
SEAL++	**0.809**	**0.768**	**0.756**	**0.744**	**0.743**	**0.733**

In conclusion, the performance of generative retrieval models in dynamic corpora reveals certain limitations, particularly when it comes to adapting to newly added documents. For retrieving initial documents, generative models perform acceptably, with only slight degradation as the corpus grows. Models like DSI-SE and NCI show only slight performance drops, similar to traditional methods like BM25 and DPR. However, the challenge arises when retrieving

newly added documents. Generative models that rely on numeric-based docids (e.g., DSI-SE and Ultron-PQ) struggle significantly, as they cannot adapt to new documents without retraining. In contrast, models using text-based docids, like SEAL and LTRGR, perform better, as text-based docids can generalize and adapt to new documents more effectively. While text-based docids show promise, not all text-based generative retrieval models perform equally well. For instance, models like SEAL demonstrate better performance compared to others like Ultron-URL. Further exploration is needed to determine which specific text features are most suitable for dynamic corpora scenario.

5 Analysis of Docid Design

To understand the pros and cons of different docid designs for GR in dynamic corpus scenarios, we conduct a series of experiments. In Section 5.1, we introduce the Initial Document Bias Index (IDBI) to analyze the bias of different GR methods towards older documents when new documents are added. We observe that numeric-based docids exhibit significantly higher bias compared to text-based docids, which, to some extent, explains the poor performance of these methods in dynamic corpus settings. Then, in Section 5.2, we conduct a comprehensive ablation study on text-based docid methods to investigate how docid type, granularity, and vocabulary size affect models' generalization to new documents. Our analysis shows that more semantic, finer-grained, and larger vocabulary choices often lead to superior results.

5.1 Bias to initial documents

To explain the poor performance of numeric-based docids on new documents, we hypothesize as follows:

Hypothesis 1 (Semantic Familiarity). *The effectiveness of generative retrieval (GR) on new documents is correlated with how well the docid representations align with the language model's pretraining distribution. Formally, let $P_{LM}(x)$ denote the token distribution learned by the language model over the vocabulary $\mathcal{V}$, and let $P_{docid}(x)$ represent the probability distribution induced by the docid representation space. We define the semantic familiarity of a docid system as:*

$$\mathcal{S} = \mathbb{E}_{x \sim P_{docid}} \left[\log P_{LM}(x) \right]. \tag{6}$$

A higher value of $\mathcal{S}$ indicates better alignment between the docid distribution and the language model's pretraining distribution.

Text-based docids inherently preserve distributional alignment with the underlying language model. By leveraging substrings (e.g., n-grams) extracted from document text, these identifiers ensure lexical overlap between new documents and the model's pretraining data, thereby maintaining high semantic familiarity.

In contrast, numeric-based docids, despite capturing latent semantic structures through clustering or vector quantization, form a distinct symbolic system that lacks grounding in the model's pretraining data. This misalignment impairs the model's ability to associate a new document with its identifier, particularly in scenarios where the document itself has never been encountered before. As a result, generative retrieval systems exhibit retrieval bias, favoring initial documents over newly introduced ones.

To quantify the retrieval bias toward initial documents, we introduce the **Initial Document Bias Index (IDBI)**.

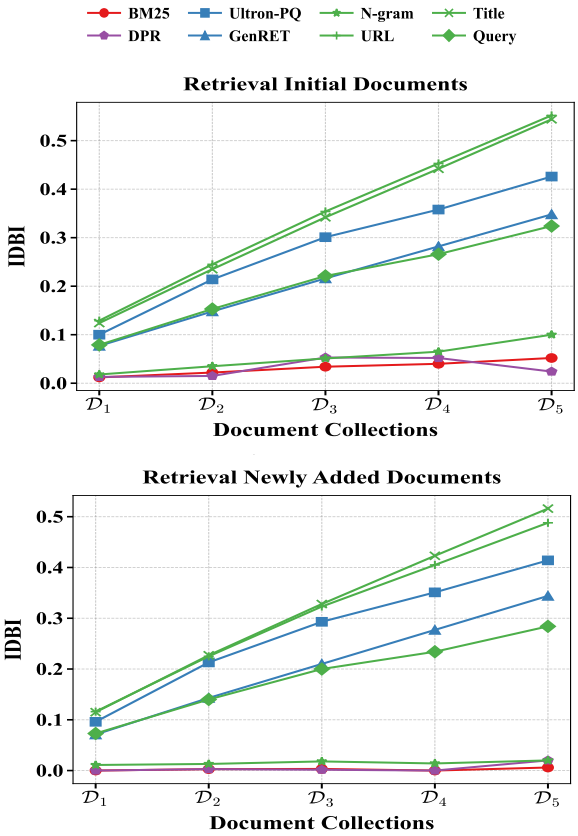

Figure 1: IDBI results for different retrieval methods on NQ dataset. Lower is better.

Definition 3 (Initial Document Bias Index (IDBI)). *The **IDBI** measures the discrepancy between the observed proportion of initial documents retrieved in the Top-K results and their expected proportion under an unbiased distribution. It is defined as:*

$$IDBI = \frac{R_{init} - E_{init}}{K - E_{init}}, \tag{7}$$

where: R_{init} is the number of retrieved initial documents in the Top-K results. $E_{init} = K \times \frac{|\mathcal{D}_0|}{|\mathcal{D}_0 \cup \mathcal{D}_{new}|}$ is the expected count of initial documents under a uniform distribution.

Remark. The IDBI assumes that initial documents $\mathcal{D}_0$ and new documents $\mathcal{D}_{\text{new}}$ are drawn from the same latent query-document relevance distribution. The index is normalized within the range $[0, 1]$, where: IDBI = 0 indicates no bias (i.e., retrieved documents are proportional to their corpus distribution). IDBI = 1 indicates complete bias toward initial documents.

As shown in Figure 1, numeric-based docids (e.g., GenRET, Ultron-PQ) exhibit significant retrieval bias toward initial documents across both task settings. This aligns with our hypothesis that numeric-based representations introduce a semantic gap, making it harder for the model to associate new documents with their identifiers. While text-based docids (e.g., URL, title, query) demonstrate better semantic alignment, they still exhibit retrieval bias, particularly for titles and URLs. This is likely due to their abstract nature, which fails to capture fine-grained document content. In contrast, BM25 and

Table 4: Comparison of retrieval performance (Hit@10) on the NQ dataset for initial and newly added documents across various text-type docid representations.

Text Type	NQ (Hit@10)					
	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$
Initial documents						
n-gram	0.753	0.712	0.688	0.666	0.642	0.626
url	**0.759**	**0.757**	**0.757**	**0.754**	**0.754**	**0.752**
title	0.633	0.630	0.615	0.588	0.579	0.558
query	0.664	0.651	0.647	0.645	0.644	0.637
Newly added documents						
n-gram	0.753	**0.695**	**0.677**	**0.632**	**0.627**	**0.607**
url	**0.759**	0.325	0.296	0.263	0.208	0.168
title	0.633	0.362	0.341	0.336	0.322	0.308
query	0.664	0.341	0.335	0.312	0.318	0.306

DPR, which perform direct text similarity matching without relying on predefined docids, show minimal retrieval bias. Their ability to retrieve new documents equitably stems from their reliance on content-based representations rather than fixed identifier mappings. N-gram docids achieve comparable performance by structurally aligning with the generative model's learning paradigm. Unlike titles or URLs, which require abstract semantic interpretation, n-grams preserve raw token sequences, which the model inherently optimizes for during generation. This alignment ensures that even previously unseen documents benefit from the model's pre-existing familiarity with local linguistic patterns.

5.2 Ablation study of text-based docid

To examine the key factors of text-based docid design, we conduct an ablation study across three aspects: (i) Docid type, where we compare different docid types, such as title, URL, n-grams, etc. (ii) Docid granularity, where we compare docids defined at different levels of granularity, including document, paragraph, sentence, and n-gram levels. (iii) Docid lexical diversity, where we compare the number of possible tokens used in different docid designs.

Docid type. First, we examine various forms of text-based docids, including URLs, titles, pseudo queries, and n-grams, to evaluate their effectiveness over dynamic corpora.

Table 4 shows the results on NQ dataset for retrieval on initial documents and new documents. We observe that the n-gram representation of docids outperforms other text-based representations with newly added documents, and also shows competitive results for initial documents in dynamic corpora. Furthermore, this advantage stems from the inherent alignment between the n-gram representation and the pre-training objectives of language models, as both explicitly model local semantic structures through sequential token prediction, enabling n-gram docids to better adapt to the model's retrieval behavior in dynamic corpora.

Docid granularity. We can define text-based docids at different levels of granularity. For example, when using the title, we essentially define the docid at the document level, where a single docid represents the entire document. In contrast, for an n-gram-based docid, any substring at any position within the document can serve as a docid to characterize the document, making it a special case

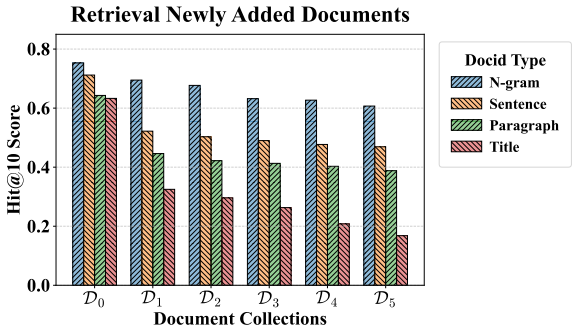

Figure 2: Hit@10 performance of n-gram docid and three fixed-position docid approaches on the NQ dataset.

Table 5: Dimensional comparison n-gram vs. other types.

Docid type	Title	URL	N-gram	Numeric
Size	5,736	17,195	53,544	32–10,000

of a *multi-docid* design. A finer-grained docid design may enable more nuanced matching between queries and documents and provides greater flexibility in docid generation strategies. But it also introduces a higher memorization overhead for the retrieval model.

To evaluate the impact of different docid granularities on model generalization, we define various text-based docids as follows: (i) *Document-level*: The title is used as an abstract representation of the document. (ii) *Paragraph-level*: Complete paragraphs serve as docids, and the document ranking is determined by the highest-ranked retrieved paragraph. (iii) *Sentence-level*: Similar to the paragraph-level approach, but each sentence is treated as a retrieval unit. (iv) *N-Gram-level*: The most fine-grained docid design, where any continuous text span within a document is used as a docid. We adopt the training and inference techniques proposed in SEAL.

Figure 2 demonstrates that the n-gram docid approach achieves the highest Hit@10 performance, significantly outperforming the document-level (title), paragraph-level, and sentence-level baselines. This result highlights the effectiveness of the fine-grained n-gram-based multi-docid design, which enables more flexible and precise query-document matching.

Docid lexical diversity. Another important factor of text-based docids is lexical diversity. Despite language models using a fixed-size vocabulary, different docids usually exhibit varying levels of lexical diversity. For example, for title-based docids, the model is forced to memorize document-docid mappings rather than generalize. When using titles as docids, the model associates specific phrases (e.g., "Climate Change Impacts") with fixed document identifiers. During inference, this rigid mapping fails when new documents introduce title variants (e.g., "Global Warming Effects"). In contrast, the high-dimensional space of n-gram docids encourages the model to learn distributional patterns rather than discrete mappings. By exposing the model to diverse substrings during training, it becomes more robust to lexical variations in new documents.

To measure the lexical diversity of different docids, we compute the *effective vocabulary size*, i.e., the number of unique words utilized by the docids of all documents in the corpus. As quantified in Table 5, n-gram docids use effective vocabulary sizes that are 10–100× larger than Title or URL docids and 100–1000× larger than

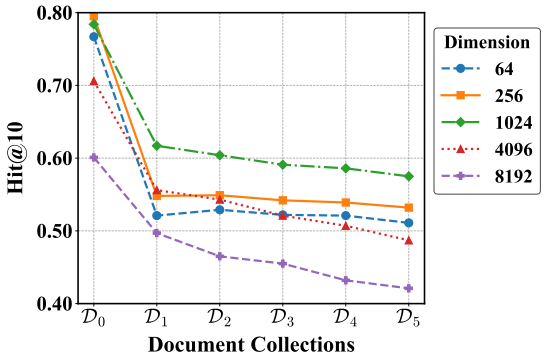

Figure 3: Hit@10 performance of different decoding dimension of Ultron-PQ on NQ dataset

numeric-based docids. This expanded dimensional space ensures that even high-frequency n-grams in initial documents occupy only a small fraction of the total capacity, forcing the model to learn generalized substring generation patterns rather than memorizing fixed mappings.

6 Enhancing Numeric Docids

While text-based docids (e.g., n-grams) exhibit superior generalization in dynamic corpora, they have several limitations: (i) high computational overhead due to complex constrained decoding (e.g., FM-index traversal for n-gram matching), (ii) memory inefficiency from storing massive text fragments, and (iii) incompatibility with non-textual data (e.g., tables, images). Conversely, numeric-based docids offer inherent advantages in storage efficiency (fixed-length numbers) and decoding speed (simple token generation), but their fixed semantic encoding limits dynamic adaptability. This leads to the natural question of how to combine the advantages of both methods to design a docid scheme that is both efficient and effective.

Previous analyses highlight several key factors for improving GR performance in dynamic corpus scenarios: (i) The docid should consider the *semantic familiarity* to the GR model and avoid using overly unfamiliar docids. (ii) The docid can be designed with finer granularity. (iii) The docid can be designed with higher lexical diversity.

Based on these insights, we explore techniques to improve *numeric-based* docid approaches. Our methods focus on three key aspects: (i) We investigate how vocabulary size influences the performance of numeric-based docids and conclude that a larger vocabulary size leads to improved performance. (ii) We explore a finer-grained docid design by assigning a docid to each chunk of a document rather than a single docid to the entire document. (iii) We address the semantic familiarity problem by assigning known docids to new documents and introducing a novel inference strategy to enhance the model's ability to generalize to unseen documents.

Combining these three aspects, we propose the **Multi-Docid Generative Retrieval (MDGR)** framework, a novel numeric-based docid design that retains the efficiency advantages of previous approaches while addressing their limitations in dynamic corpora scenarios.

6.1 Docid design of MDGR

Optimizing vocabulary size. Building on the insights from analysis of lexical diversity of text-based docid, we hypothesize that expanding the decoding dimension for number-type docids in GR models could similarly influence their adaptability to dynamic corpora. To test this hypothesis, we adapt the Ultron-PQ framework by modifying its PQ parameters - the number of clusters in each subspace. In PQ-based docid generation, document embeddings are partitioned into m subvectors, each quantized into k clusters via k-means. By varying k while fixing m, we systematically control the decoding dimension: our experiments explore $k \in \{64, 256, 1024, 4096, 8192\}$ with $m = 4$.

As shown in Figure 3, the experimental results highlight two key patterns in docid dimension scaling. First, expanding the docid size from $k = 64$ to $k = 1024$ shows minimal impact on initial document retrieval but substantially improves performance on new documents. Second, excessive dimensions ($k = 4096, 8192$) cause sharp declines across all test sets. This could due to oversized size create sparse, under-trained docid mappings, which cause the model fails to establish reliable semantic-code relationships, particularly for new documents. Based on these findings, we use docid size of $k = 1024$ for our following study.

Designing fine-grained docids. We explore a multi-docid approach to designing numeric-based docids with finer granularity. Specifically, we partition a document d into semantic chunks $\{c_1, c_2, \ldots, c_N\}$ using a sliding window approach, where each c_i represents a text fragment of the document. Each chunk is independently mapped to a numeric-based docid z_i through product quantization (PQ). This results in multiple number sequences per document, analogous to the multi-docid design of n-grams.

Constrained docid expansion. To address the semantic gap in numeric-based docids, we propose a constrained docids expansion strategy on dynamic corpora. We store all docids from the initial document collection, and constrain the indexer to use only these existing docids when indexing new document chunks.

6.2 Inference strategy of MDGR

To retrieve the relevant documents, we propose a simple document ranking strategy for our approach. Given a query, we first perform constrained beam search to generate several candidate docids $\{z_1, z_2, \ldots, z_k\}$. We then retrieve all documents containing at least one of these docids. For each candidate document d_j, we compute a score based on the weighted sum of the beam search positions of the contained docids. The score for document d_j is given by:

$$\text{Score}(d_j) = \text{Coverage Count}(d_j) + \beta \times \sum_{z_i \in d_j} \frac{1}{\text{rank}(z_i)}, \quad (8)$$

where Coverage Count(d_j) is the number of distinct docids in d_j; rank(z_i) is the position of docid z_i in the beam search (with 1 being the highest rank); and β is a hyperparameter that controls the importance of the ranking term relative to the coverage count.

6.3 Implementation and evaluation results

Our model is implemented using the T5-base architecture. We initialize the model with pre-trained weights from Hugging Face's

Table 6: Comparison of retrieval performance (Hit@10) on the NQ dataset for newly added documents.

Method	NQ (Hit@10)					
	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$
BM25	0.647	0.620	0.588	0.598	0.552	0.571
DPR-HN	0.826	0.645	0.644	0.626	0.621	0.624
NCI	**0.871**	0.464	0.437	0.433	0.358	0.323
SEAL	0.809	**0.744**	**0.736**	**0.727**	**0.727**	**0.725**
MDGR	0.824	0.717	0.704	0.695	0.647	0.633
Ablation study						
w/o constrain	0.824	0.447	0.424	0.408	0.377	0.356
w/o multi-docid	0.831	0.523	0.511	0.497	0.488	0.473
MDGR (size = 64)	0.843	0.472	0.436	0.417	0.409	0.381
MDGR (size = 8192)	0.674	0.574	0.565	0.533	0.531	0.512

T5-base checkpoint. Following previous work, the training objective includes three parts: (i) **Training with synthetic queries.** Minimize the loss between generated pseudo-queries and their corresponding docids. (ii) **Encoding document contexts.** Minimize the loss between each document chunk c_i and its original docid. (iii) **Incorporating real queries.** Minimize the loss between user queries and target docids.

For training, we employ the AdamW optimizer with a base learning rate of 1×10^{-5} and a linear warmup over the first 10% of training steps. We set the batch size to 64 and train for 10 epochs, which requires approximately 4 hours on 8 NVIDIA A100 GPUs. To ensure efficient document processing, we split documents into chunks using a sliding window approach: each chunk contains 256 tokens with a stride of 128 tokens. To get the numeric-based docids, we first generate semantic embeddings for text chunks using a frozen BERT-base model, then apply product quantization to convert these continuous vectors into docids. The docids have a size of 1024 and a length of 4.

As shown in Table 6, the MDGR framework exhibits competitive effectiveness in comparison to existing retrieval models. While the performance of MDGR is not the absolute best across all document sets, it achieves solid results over dynamic corpora.

We also show some ablation variant to evaluate three critical design of our method: First, removing the constrained docid expansion strategy leads to significant deterioration, demonstrating that generating new docids for updated documents would disrupt the semantic consistency of numbering-based docids. Second, when disabling multi-docid indexing (single-docid per document), the performance drops significantly, confirming our hypothesis that fine-grained docid design can better preserves granular semantic associations. Furthermore, we investigate the impact of docid size. Small docid sizes (64) lead to substantially degrade on newly added documents. Conversely, excessively large sizes (8192) create an over-discretized learning space where the model struggles to establish stable query-docid mappings.

Table 7 shows that MDGR achieves both effectiveness and system efficiency. This efficiency stems from combining number-type docids' compact storage with number sequences, enabling dynamic adaptability without sacrificing speed – MDGR retrieves documents 3.6× faster than SEAL while outperforming other numeric-based docid methods (NCI) in dynamic corpora scenario.

Table 7: Experiments about memory costs and efficiency.

Method	Memory	Tok-K	Latency
DPR	980MB	100	152ms
NCI	865MB	10	216ms
		100	269ms
SEAL	2200MB	10	619ms
		100	778ms
MDGR	886MB	10	241ms
		100	320ms

7 Conclusion

In this paper, we have conducted a systematic investigation into the capabilities and limitations of GR models in dynamic corpora scenarios, where document collections expand continuously over time. Through comprehensive evaluations on realistic dynamic benchmarks derived from NQ and MS-MARCO datasets, we have revealed that certain GR methods, primarily those relying on numeric-based docids, face challenges in generalizing to unseen documents without additional training, while models utilizing text-based docids demonstrate stronger generalization capabilities in dynamic corpora. Our analysis has demonstrated that text-based docids inherently address key limitations of numeric-based docids by mitigating generation bias towards previously seen docids, enabling finer-grained document representation, and enhancing robustness against overfitting.

Building on these insights, we have proposed a novel GR framework MDGR that combines the structured benefits of numeric-based docids with a revised design to avoid training-induced biases, achieving competitive performance in dynamic corpora scenario. Our findings not only highlight the critical role of docid semantics and representation in GR frameworks but also provide actionable guidelines for adapting generative retrieval to real-world dynamic environments. Future work may explore hybrid docid strategies or pretraining-enhanced generalization mechanisms to further bridge the gap between static and dynamic retrieval performance.

Acknowledgments

This research was (partially) funded by the Natural Science Foundation of China (62472261), the Provincial Key R&D Program of Shandong Province with grant No. 2024CXGC010108, the Technology Innovation Guidance Program of Shandong Province with grant No. YDZX2024088, the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and the European Union's Horizon Europe program under grant agreement No 101070212. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016.

Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).

[2] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems* 35 (2022), 31668–31683.

[3] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 306–315.

[4] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 191–200.

[5] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904* (2020).

[6] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, Jiafeng Guo, et al. 2022. Pre-training methods in information retrieval. *Foundations and Trends® in Information Retrieval* 16, 3 (2022), 178–317.

[7] Jiafeng Guo, Changjiang Zhou, Ruqing Zhang, Jiangui Chen, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024. Corpusbrain++: A continual generative pre-training framework for knowledge-intensive language tasks. *arXiv preprint arXiv:2402.16767* (2024).

[8] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*. PMLR, 2790–2799.

[9] Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, et al. 2023. Language models as semantic indexers. *arXiv preprint arXiv:2310.07815* (2023).

[10] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[11] Chaeeun Kim, Soyoung Yoon, Hyunji Lee, Joel Jang, Sohee Yang, and Minjoon Seo. 2023. Exploring the practicality of generative retrieval on dynamic corpora. *arXiv preprint arXiv:2305.18952* (2023).

[12] Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q Weinberger. 2023. Incdsi: incrementally updatable document retrieval. In *International Conference on Machine Learning*. PMLR, 17122–17134.

[13] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[14] Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2024. From matching to generation: A survey on generative information retrieval. *arXiv preprint arXiv:2404.14851* (2024).

[15] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview identifiers enhanced generative retrieval. *arXiv preprint arXiv:2305.16675* (2023).

[16] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8716–8723.

[17] Yongqi Li, Zhen Zhang, Wenjie Wang, Liqiang Nie, Wenjie Li, and Tat-Seng Chua. 2024. Distillation Enhanced Generative Retrieval. *arXiv preprint arXiv:2402.10769* (2024).

[18] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: An easy-to-use python toolkit to support replicable ir research with sparse and dense representations. *arXiv preprint arXiv:2102.10073* (2021).

[19] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2022. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature.

[20] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Pre-train a discriminative text encoder for dense retrieval via contrastive span prediction. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 848–858.

[21] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Scattered or connected? an optimized parameter-efficient tuning approach for information retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1471–1480.

[22] Christopher D. Manning, Hinrich Schütze, and Prabhakar Raghavan. 2009. *Introduction to Information Retrieval*. Cambridge University Press.

[23] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. 2022. DSI++: Updating transformer memory with new documents. *arXiv preprint arXiv:2212.09744* (2022).

[24] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. In *ACM SIGIR Forum*, Vol. 55. ACM New York, NY, USA, 1–27.

[25] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191* (2020).

[26] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[27] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. 2024. Learning to tokenize for generative retrieval. *Advances in Neural Information Processing Systems* 36 (2024).

[28] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024. Generative Retrieval Meets Multi-Graded Relevance. *arXiv preprint arXiv:2409.18409* (2024).

[29] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024. Listwise generative retrieval models via a sequential learning process. *ACM Transactions on Information Systems* 42, 5 (2024), 1–31.

[30] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems* 35 (2022), 21831–21843.

[31] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems* 35 (2022), 25600–25614.

[32] Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2023. Auto search indexer for end-to-end document retrieval. *arXiv preprint arXiv:2310.12455* (2023).

[33] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024. Scalable and effective generative information retrieval. In *Proceedings of the ACM on Web Conference 2024*. 1441–1452.

[34] Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. 2024. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 458–468.

[35] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing generative retrieval with reinforcement learning from relevance feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 12481–12490.

[36] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Yu Wu, Peitian Zhang, and Ji rong Wen. 2022. Ultron: An Ultimate Retriever on Corpus with a Model-based Indexer. ArXiv abs/2208.09257.

[37] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128* (2022).

[38] Noah Ziems, Wenhao Yu, Zhihan Zhang, and Meng Jiang. 2023. Large language models are built-in autoregressive search engines. *arXiv preprint arXiv:2305.09612* (2023).